1 Objectives

It is possible to use functions to represent the *control flow* of a program. This technique is called *continuation passing style*. After today's lecture, you should be able to

- explain what CPS is,
- give an example of a programming technique using CPS, and
- transform a simple function from direct style to CPS.

2 Examples

```
# let gcdstar lst k =
    let rec aux lst newk = match lst with
        [] -> newk 0
      | 1::xs -> k 1
      | x::xs -> aux xs
                      (fun res -> newk (gcd x res))
    in aux lst k;;
val gcdstar : int list -> (int -> 'a) -> 'a = <fun>
# gcdstar [44;12;80;6] report;;
2
# gcdstar [44;12;1;80;6] report;;
1
```

3 Problems

Try the following problems. In a few minutes the instructor will go over the solutions. Feel free to work with the person next to you!

- The `gcdstar` example didn't go as far as it could have, because the function gcd was left in direct style. Transform `gcd` into CPS, and then write the `gcdstar` function in CPS to use it. It will look very similar to the lecture CPS version of `gcdstar`.
- Suppose now we want to perform multiplications, and we want to do them after we've done all the additions and subtractions. Write the necessary modifications.
- Now suppose we want to do an early abort if we detect a multiply by zero. How do you do that?